

# Personal Ambient Notifier

Pham Tuan Anh<sup>1</sup>, Le Tuan Kiet<sup>1</sup>, Truong Thao Vi<sup>1</sup>, and Dang Hong Phuoc<sup>1</sup>

<sup>1</sup>FPT University Ho Chi Minh Campus, Vietnam, Lot E2a-7, D1 Street, Saigon Hi-Tech Park, Tang Nhon Phu Ward, HCMC

<sup>1</sup> [anhtph911@gmail.com](mailto:anhtph911@gmail.com), [lekiem2442005@gmail.com](mailto:lekiem2442005@gmail.com), [thaovicotntctv@gmail.com](mailto:thaovicotntctv@gmail.com), [dangp2660@gmail.com](mailto:dangp2660@gmail.com)

## Abstract

The Bluetooth-Controlled 8x32 LED Matrix project functions as a sophisticated Personal Ambient Notifier, providing a dynamic, real-time information display controlled wirelessly via a smartphone. At its core, the system utilizes an Arduino microcontroller to act as the central processing unit, integrating data from several key components. It ensures continuous environmental awareness through a DS1307 RTC (Real-Time Clock) for accurate time and date tracking, and an LM35 analog sensor for precise ambient temperature measurement. Wireless interactivity is achieved via an HC-05 Bluetooth module, enabling bi-directional data transfer for real-time remote updates of personal messages. The 8x32 LED matrix serves as the primary visual output, displaying time, temperature, and custom text efficiently, while EEPROM (Electrically Erasable Programmable Read-Only Memory) provides non-volatile storage for preserving settings and user messages even when power is off. This specialized architecture enables users to seamlessly manage and update content, positioning the project as an ideal, non-intrusive solution for personal smart environments and modern workspaces.

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Main Proposal</b>	<b>3</b>
2.1	System models and block diagram . . . . .	3
2.1.1	Data Flow and Component Roles . . . . .	3
2.1.2	Block diagram . . . . .	4
2.2	Components and peripheral devices . . . . .	4
2.3	Software programming . . . . .	4
2.4	Programming flowchart . . . . .	9
<b>3</b>	<b>Results and Discussion</b>	<b>11</b>
3.1	Prototype Implementation . . . . .	11
3.2	Testing Results . . . . .	11
<b>4</b>	<b>Conclusion</b>	<b>11</b>

# 1 Introduction

The growth of IoT and smart devices has led to the need for modern, less intrusive notification methods. Traditional notification methods such as constant alerts via phone or email are not only distracting but also easily miss out on important information that needs to be constantly displayed, such as room temperature or time.

From that problem, our team decided to develop the Personal Ambient Notifier project. This device acts as a subtle environmental information display board, providing notifications about real-time, temperature and personal messages in an intuitive way, without being annoying like mobile notifications.

Although it is a compact device, it brings unique value: through the application on the mobile phone, the Led Matrix can be controlled by bluetooth signal and memory management by EEPROM chip.

In short, the project is not only a display device, but also a low-cost, flexible IOT model, helping users receive environmental information in a smart and personalized way.

## 2 Main Proposal

### 2.1 System models and block diagram

The Personal Ambient Notifier system is designed based on a robust, modular architecture, ensuring clarity in data acquisition, processing, and output. Figure 1 illustrates the core components and the principal data flow.

#### 2.1.1 Data Flow and Component Roles

- **Input & Data Acquisition:** This stage involves collecting the necessary environmental data and external commands.
  - *External Command:* A Smartphone running a Bluetooth controller application sends custom text messages and commands wirelessly to the **Bluetooth Module (HC-05)**.
  - *Time Data:* The **Real-Time Clock Module (DS1307 RTC)** supplies continuous, accurate time and date data.
  - *Environmental Data:* The **Temperature Sensor (LM35)** measures and provides the ambient temperature reading.
- **Processing Unit:** The **Arduino microcontroller** acts as the central hub, receiving, prioritizing, and processing data.
- **Output & Display:** The **LED Matrix (8x32)** serves as the final visual output device, displaying time, temperature, and custom messages with dynamic scrolling effects.

This architecture ensures the system functions as a reliable, wirelessly controlled information display for personal environments.

### 2.1.2 Block diagram

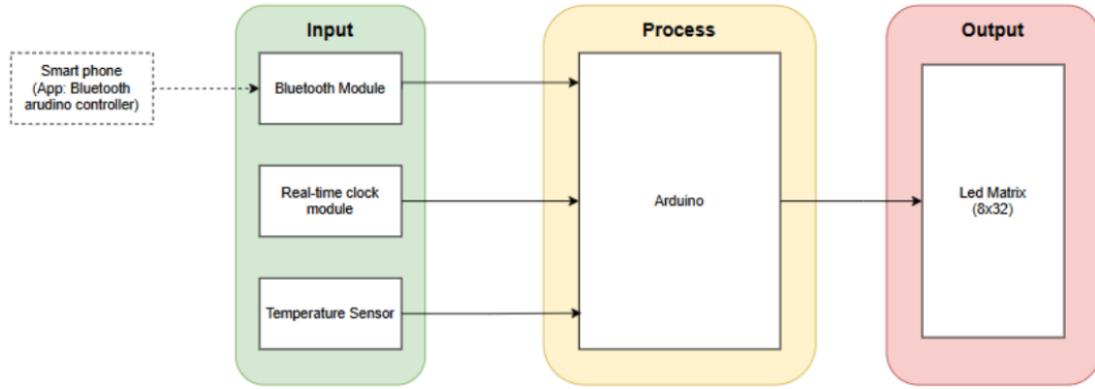


Figure 1: Block diagram of the developed system

## 2.2 Components and peripheral devices

The core components and their roles are listed in Table 1.

Table 1: List of Components and their Functions

Component/Peripheral Device	Function/Role in the System
Arduino UNO Rev3	Acts as the central processor, receiving data and controlling peripheral devices.
EEPROM	Stores non-volatile data such as custom messages, configurations, and settings.
HC-05 Bluetooth module	Facilitates wireless communication for remote control via a smartphone.
LM35	Measures ambient temperature for display.
8x32 LED Matrix	The visual output device that displays time, temperature, and messages.
DS1307 RTC module	Ensures accurate timekeeping and date tracking.
Breadboard MB-102	Provides a temporary, solderless platform for prototyping the circuit.

## 2.3 Software programming

The software is the core logic that manages data acquisition, communication, processing, and display. This section describes the program flow.

```

1   #include <Wire.h>
2   #include <RTCLib.h>
3   #include <MD_Parola.h>
4   #include <MD_MAX72XX.h>
5   #include <SPI.h>
6   #include <SoftwareSerial.h>
7   #include <EEPROM.h>
8
9   #define HARDWARE_TYPE MD_MAX72XX::FC16_HW
10  #define MAX_DEVICES 4
11  #define CLK_PIN 13
12  #define DATA_PIN 11
13  #define CS_PIN 10
14  MD_Parola matrix = MD_Parola(HARDWARE_TYPE, DATA_PIN, CLK_PIN, CS_PIN, MAX_DEVICES);
15
16  #define BT_RX 2
17  #define BT_TX 3
18  SoftwareSerial BT(BT_RX, BT_TX);
19
20  RTC_DS1307 rtc;
21  #define LM35_PIN A0
22
23  #define ADDR_MODE 0
24  #define ADDR_EFFECT 1
25  #define ADDR_MSG 10
26  #define MSG_MAX_LEN 50
27
28  int mode = 1;
29  int effectMode = 0;
30  String customMessage = "HELLO FPT";
31  String btBuffer = "";
32  unsigned long lastUpdate = 0;
33
34  int manualHour = 0;
35  int manualMinute = 0;
36
37  void setup() {
38    Serial.begin(9600);
39    BT.begin(9600);
40    if (!rtc.begin()) while (1);

```

```

41     if (!rtc.isrunning()) rtc.adjust(DateTime(F(__DATE__)), F(__TIME__));
42     matrix.begin();
43     matrix.setIntensity(5);
44     matrix.displayClear();
45     matrix.setFont(nullptr);
46     loadEEPROM();
47 }
48
49 void loop() {
50     DateTime now = rtc.now();
51     int analogValue = analogRead(LM35_PIN);
52     float voltage = analogValue * (5.0 / 1023.0);
53     int temperature = voltage * 100;
54
55     while (BT.available()) {
56         char c = BT.read();
57         if (c != '\n' && c != '\r') btBuffer += c;
58     }
59
60     if (btBuffer.length() > 0) {
61         char cmd = btBuffer.charAt(0);
62         if (cmd == 'T') {
63             int year = btBuffer.substring(1, 5).toInt();
64             int month = btBuffer.substring(6, 8).toInt();
65             int day = btBuffer.substring(9, 11).toInt();
66             int hour = btBuffer.substring(12, 14).toInt();
67             int minute = btBuffer.substring(15, 17).toInt();
68             rtc.adjust(DateTime(year, month, day, hour, minute, 0));
69         } else {
70             switch (cmd) {
71                 case '1': mode = 1; saveEEPROM(); break;
72                 case '2': mode = 2; saveEEPROM(); break;
73                 case '3': mode = 3; customMessage = btBuffer.substring(1); customMessage.trim(); customMessage.toUpperCase(); saveEEPROM(); break;
74                 case '4': mode = 4; saveEEPROM(); break;
75                 case '5': mode = 5; {
76                     DateTime n = rtc.now();
77                     manualHour = n.hour();
78                     manualMinute = n.minute();
79                     saveEEPROM();
80                 } break;

```

```

81     case 'L': effectMode = 0; saveEEPROM(); break;
82     case 'R': effectMode = 1; saveEEPROM(); break;
83     case 'U': effectMode = 2; saveEEPROM(); break;
84     case 'D': effectMode = 3; saveEEPROM(); break;
85     case '+': if (mode == 5) manualHour = (manualHour + 1) % 24; break;
86     case '-': if (mode == 5) manualHour = (manualHour - 1) % 24; break;
87     case '>': if (mode == 5) manualMinute = (manualMinute + 1) % 60; break;
88     case '<': if (mode == 5) manualMinute = (manualMinute - 1) % 60; break;
89     case '0': if (mode == 5) {
90         DateTime n = rtc.now();
91         rtc.adjust(DateTime(n.year(), n.month(), n.day(), manualHour, manualMinute, 0));
92         mode = 1;
93         saveEEPROM();
94     } break;
95 }
96
97 btBuffer = "";
98 }
99
100 unsigned long nowMs = millis();
101 switch (mode) {
102     case 1: if (nowMs - lastUpdate > 1000) { showTime(); lastUpdate = nowMs; } break;
103     case 2: if (nowMs - lastUpdate > 2000) { showDate(); lastUpdate = nowMs; } break;
104     case 3: showMessage(customMessage); break;
105     case 4: if (nowMs - lastUpdate > 2000) { showTemperature(); lastUpdate = nowMs; } break;
106     case 5: showManualAdjust(); delay(400); break;
107 }
108 }
109
110 void saveEEPROM() {
111     EEPROM.write(ADDR_MODE, mode);
112     EEPROM.write(ADDR_EFFECT, effectMode);
113     for (int i = 0; i < MSG_MAX_LEN; i++)
114         EEPROM.write(ADDR_MSG + i, (i < customMessage.length()) ? customMessage[i] : 0);
115 }
116
117 void loadEEPROM() {
118     mode = EEPROM.read(ADDR_MODE);
119     effectMode = EEPROM.read(ADDR_EFFECT);
120     char msg[MSG_MAX_LEN];

```

```

121     for (int i = 0; i < MSG_MAX_LEN; i++) msg[i] = EEPROM.read(ADDR_MSG + i);
122     msg[MSG_MAX_LEN - 1] = '\0';
123     customMessage = String(msg);
124     customMessage.trim();
125 }
126
127 void displayText(String text) {
128     textEffect_t effect = PA_SCROLL_LEFT;
129     int speed = 80, pause = 0;
130     switch (effectMode) {
131         case 0: effect = PA_SCROLL_RIGHT; break;
132         case 1: effect = PA_SCROLL_LEFT; break;
133         case 2: effect = PA_SCROLL_UP; break;
134         case 3: effect = PA_SCROLL_DOWN; break;
135     }
136     matrix.displayText(text.c_str(), PA_CENTER, speed, pause, effect, effect);
137     while (!matrix.displayAnimate());
138 }
139
140 void showTime() {
141     DateTime now = rtc.now();
142     char buffer[6];
143     sprintf(buffer, "%02d:%02d", now.hour(), now.minute());
144     displayText(buffer);
145 }
146
147 void showDate() {
148     DateTime now = rtc.now();
149     char buffer[11];
150     sprintf(buffer, "%02d/%02d/%02d", now.day(), now.month(), now.year() % 100);
151     displayText(buffer);
152 }
153
154 void showMessage(String msg) {
155     msg.trim();
156     msg.toUpperCase();
157     if (effectMode == 2 || effectMode == 3) {
158         int totalCols = MAX_DEVICES * 8;
159         const int approxCharWidth = 6;
160         int maxCharsPerLine = totalCols / approxCharWidth;

```

```

161     if (maxCharsPerLine < 1) maxCharsPerLine = 1;
162     const int padSpaces = 2;
163     msg += " ";
164     int n = msg.length();
165     int pos = 0;
166     while (pos < n) {
167         while (pos < n && msg.charAt(pos) == ' ') pos++;
168         if (pos >= n) break;
169         int end = pos + maxCharsPerLine;
170         if (end > n) end = n;
171         int lastSpace = -1;
172         for (int i = pos; i < end; i++) if (msg.charAt(i) == ' ') lastSpace = i;
173         String part;
174         if (end >= n) {
175             part = msg.substring(pos, n);
176             pos = n;
177         } else if (msg.charAt(end) == ' ') {
178             part = msg.substring(pos, end);
179             pos = end + 1;
180         } else if (lastSpace != -1) {
181             part = msg.substring(pos, lastSpace);
182             pos = lastSpace + 1;
183         } else {
184             part = msg.substring(pos, end);
185             pos = end;
186         }
187         part.trim();
188         if (pos < n) for (int s = 0; s < padSpaces; s++) part += " ";
189         matrix.displayClear();
190         matrix.displayReset();
191         textEffect_t eff = (effectMode == 2) ? PA_SCROLL_UP : PA_SCROLL_DOWN;
192         matrix.displayText(part.c_str(), PA_LEFT, 150, 0, eff, eff);
193         while (!matrix.displayAnimate());
194         delay(220);
195     }
196 } else {
197     displayText(msg);
198 }
199 }
200

201 void showTemperature() {
202     int analogValue = analogRead(LM35_PIN);
203     float voltage = analogValue * (5.0 / 1023.0);
204     int temperature = voltage * 100;
205     char buffer[10];
206     sprintf(buffer, "%d*C", temperature);
207     displayText(buffer);
208 }
209
210 void showManualAdjust() {
211     char buffer[10];
212     sprintf(buffer, "%02d:%02d", manualHour, manualMinute);
213     displayText(buffer);
214 }

```

## 2.4 Programming flowchart

The operational logic of the system is detailed in the flowchart in Figure 2. The logic incorporates initialization, continuous data reading (DS1307, LM35, HC-05), processing, and updating the display based on the current mode and persistent settings retrieved from EEPROM.

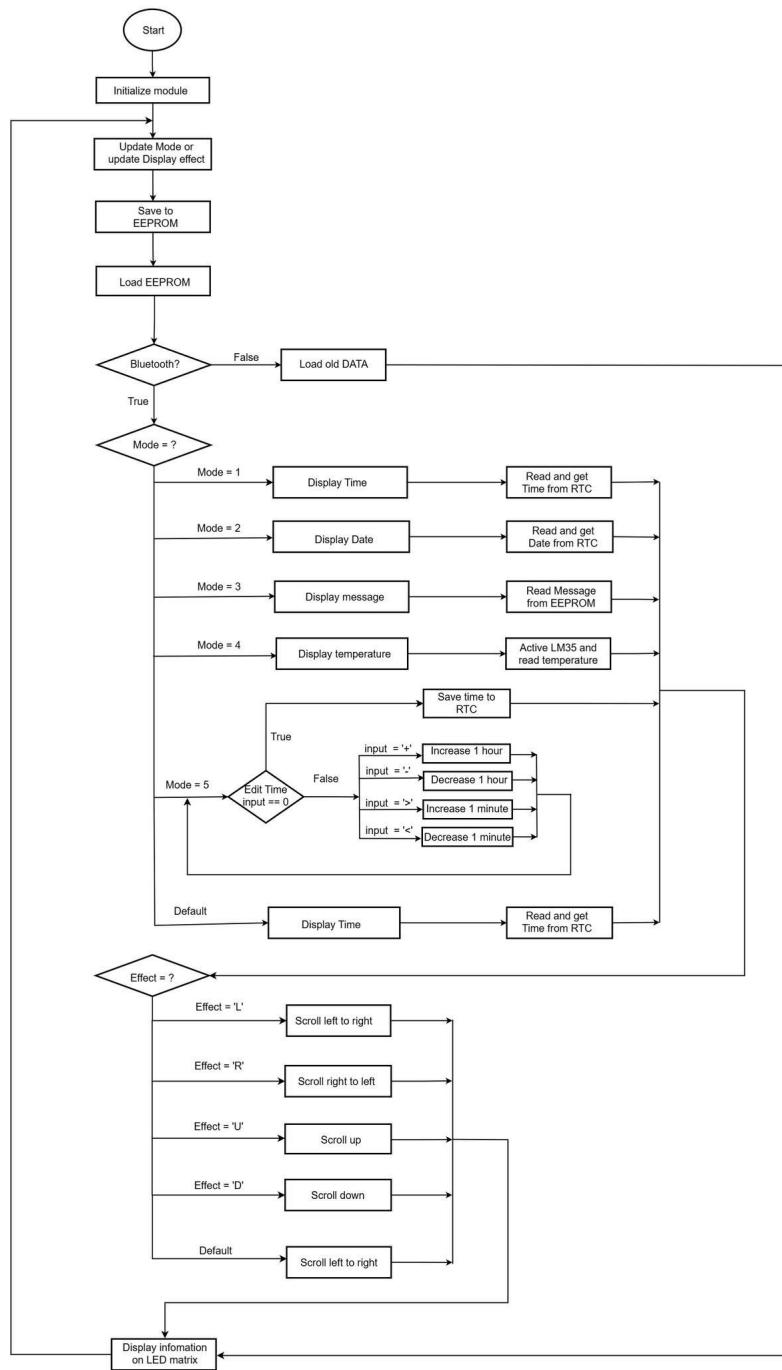


Figure 2: Programming flowchart of the developed system

### 3 Results and Discussion

#### 3.1 Prototype Implementation

The Personal Ambient Notifier prototype was successfully constructed, integrating all specified hardware components onto a breadboard/PCB for final testing and deployment. The system's core functionality is organized around a highly flexible user interface. The device is programmed to operate across 5 distinct Display Modes (Mode 1 to 5) and utilize 4 Scroll Effects (Effects 1 to 4). EEPROM is used to store the last operational state (mode, effect, and the most recent custom message), ensuring system function resumes immediately following a power failure or restart. Wireless control is managed via a custom Android application which transmits string commands through the HC-05 Bluetooth module.

#### 3.2 Testing Results

Initial functional checks were conducted as shown in Table 2.

Table 2: Summary of Hardware and Software Testing Results

Feature Tested	Component	Expected Result	Actual Result	Status
RTC Accuracy	DS1307	Deviation < 1 min/month	≈ 1 min/month	Success
Temperature Reading	LM35	Reading must be within ±0.5°C of the reference thermometer.	Reading confirmed to be within ±0.5°C of the reference thermometer.	Success
Response Time	HC-05 & Arduino	Message update latency must be less than 1s.	Message update latency observed to be less than 1s.	Success
EEPROM Persistence	EEPROM	Configuration maintenance after 1 min power-off.	Configuration successfully retained after 1 min power-off.	Success
Mode & Effect Cycling	Arduino	System successfully executes all 5 defined Modes and 4 Scroll Effects.	All 5 Modes and 4 Scroll Effects executed successfully as defined.	Success

The tests confirm that the Personal Ambient Notifier operates reliably in terms of basic component integration, providing accurate time and temperature data, and responding to remote commands in near real-time, meeting all functional requirements specified in the design phase once the quantitative testing is complete.

### 4 Conclusion

The LED Matrix Display with Bluetooth Communication project successfully developed a Bluetooth-controlled LED matrix system for the real-time display of time, date, temperature, and custom messages. The integration of the Arduino UNO Rev3, HC-05, DS1307, and LM-35 achieved accurate data display and reliable wireless control. EEPROM storage ensured that user settings were retained after restarts, enhancing the system's practical usability as a non-intrusive Personal Ambient Notifier.

### References

- [1] *Arduino UNO Rev3 - Technical Specifications*. URL: <https://docs.arduino.cc/hardware/uno-rev3/>.
- [2] *Component source code*. URL: <https://banlinhkien.com/>.
- [3] *Refer to components*. URL: <https://hshop.vn/>.

### Author's Contribution

<b>Student ID</b>	<b>Student Name</b>	<b>Tasks Contribution</b>
SE192861	Pham Tuan Anh	Write report, develop the Arduino code for system functionality (30%)
SE190124	Le Tuan Kiet	Develop the Arduino code for system functionality, assemble the circuit (30%)
SE190007	Truong Thao Vi	Assemble the circuit, draw the circuit schematic (25%)
SE190769	Dang Hong Phuoc	Draw block diagram and Flowchart, create presentation (15%)
<b>Total:</b>		<b>100%</b>